

FIG. 1

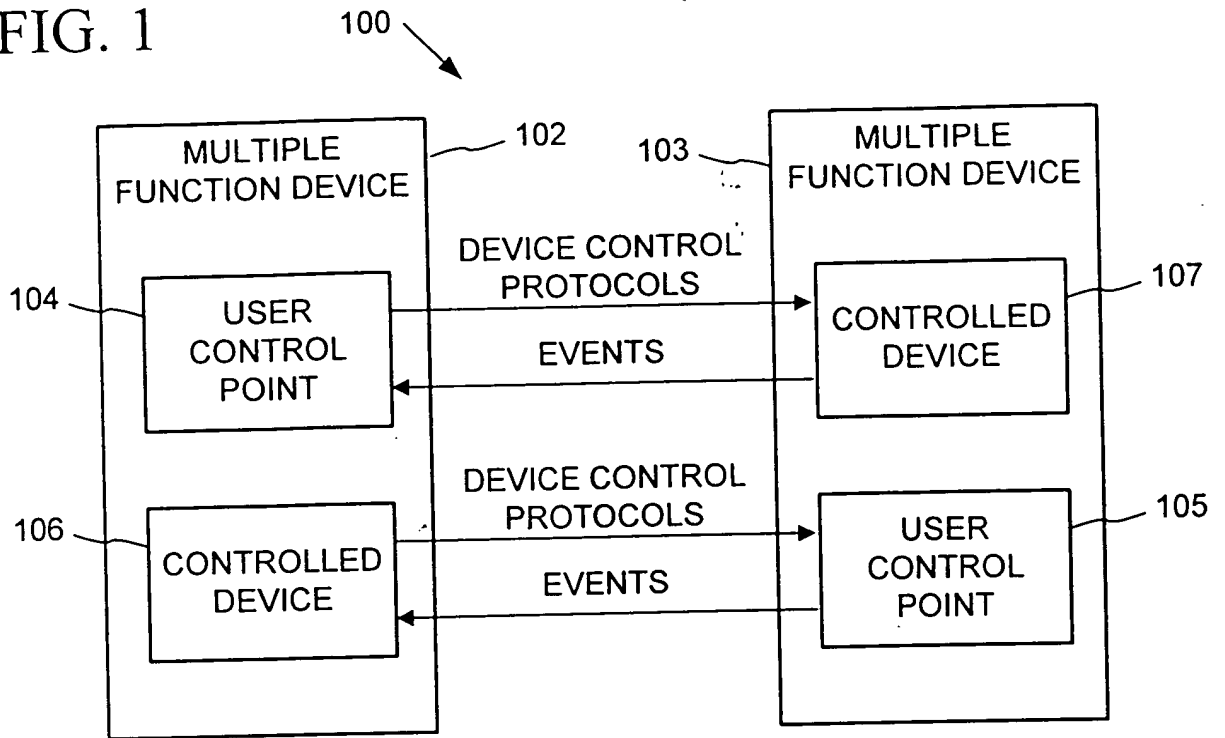


FIG. 2

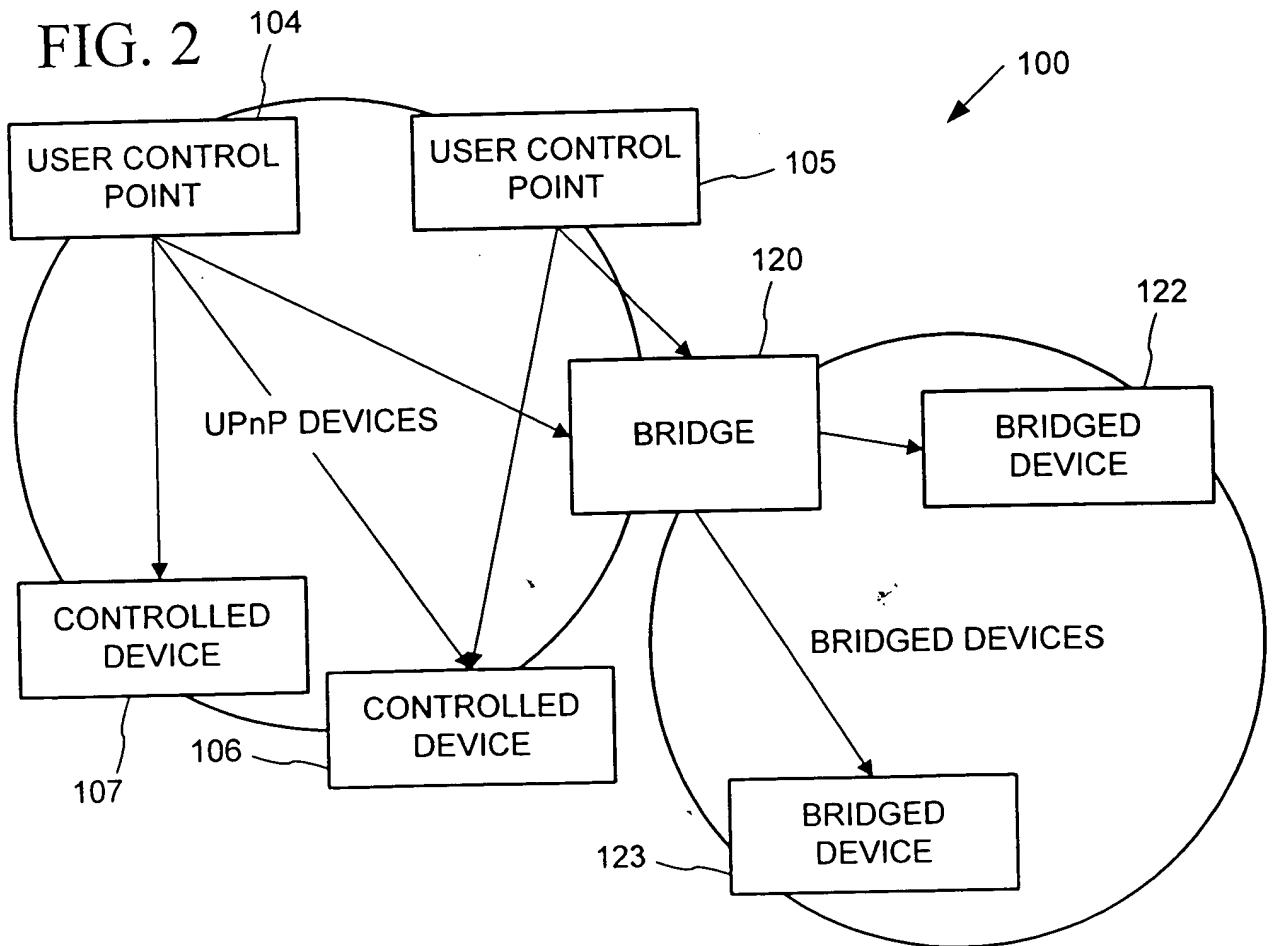


FIG. 3

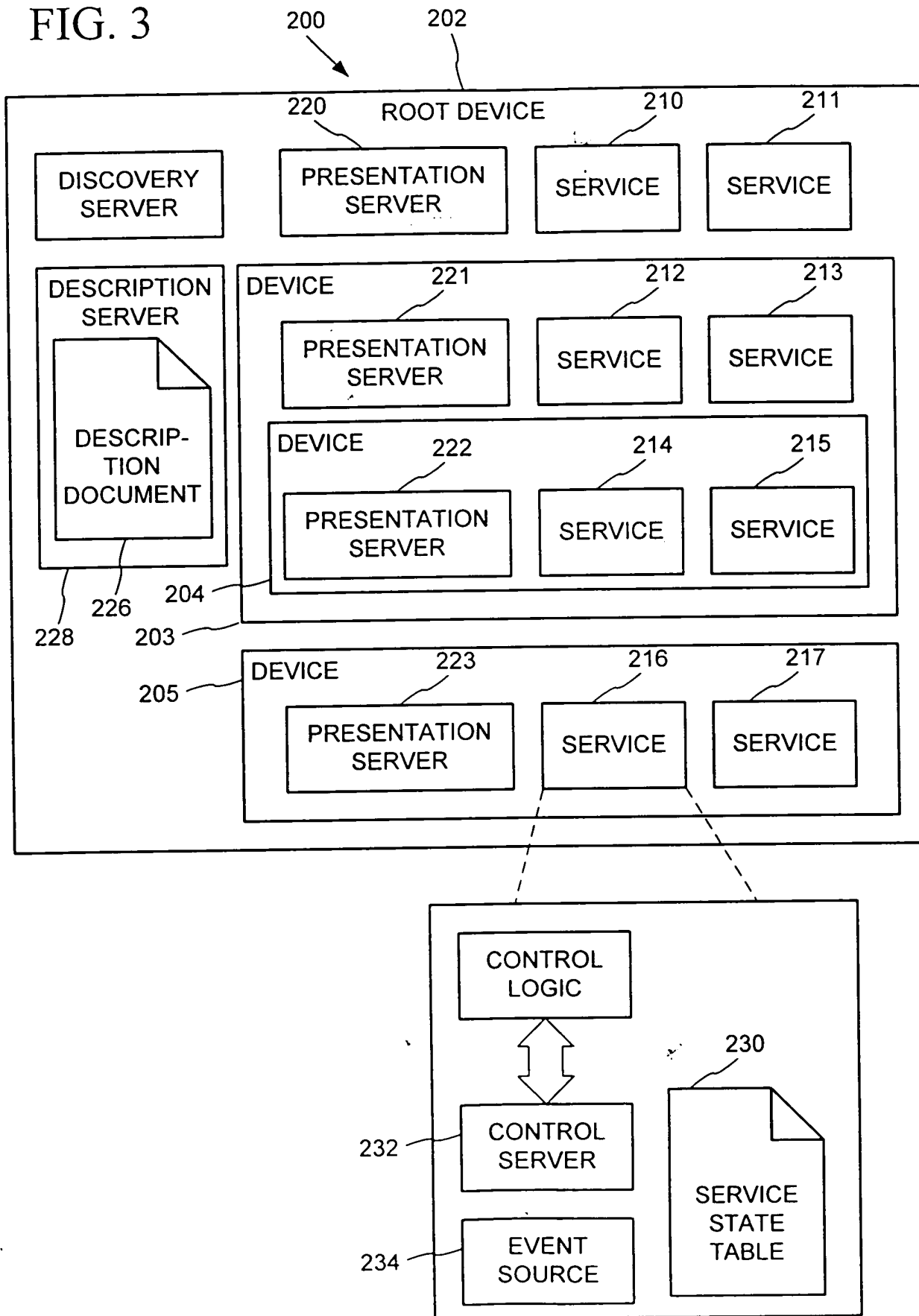


FIG. 4

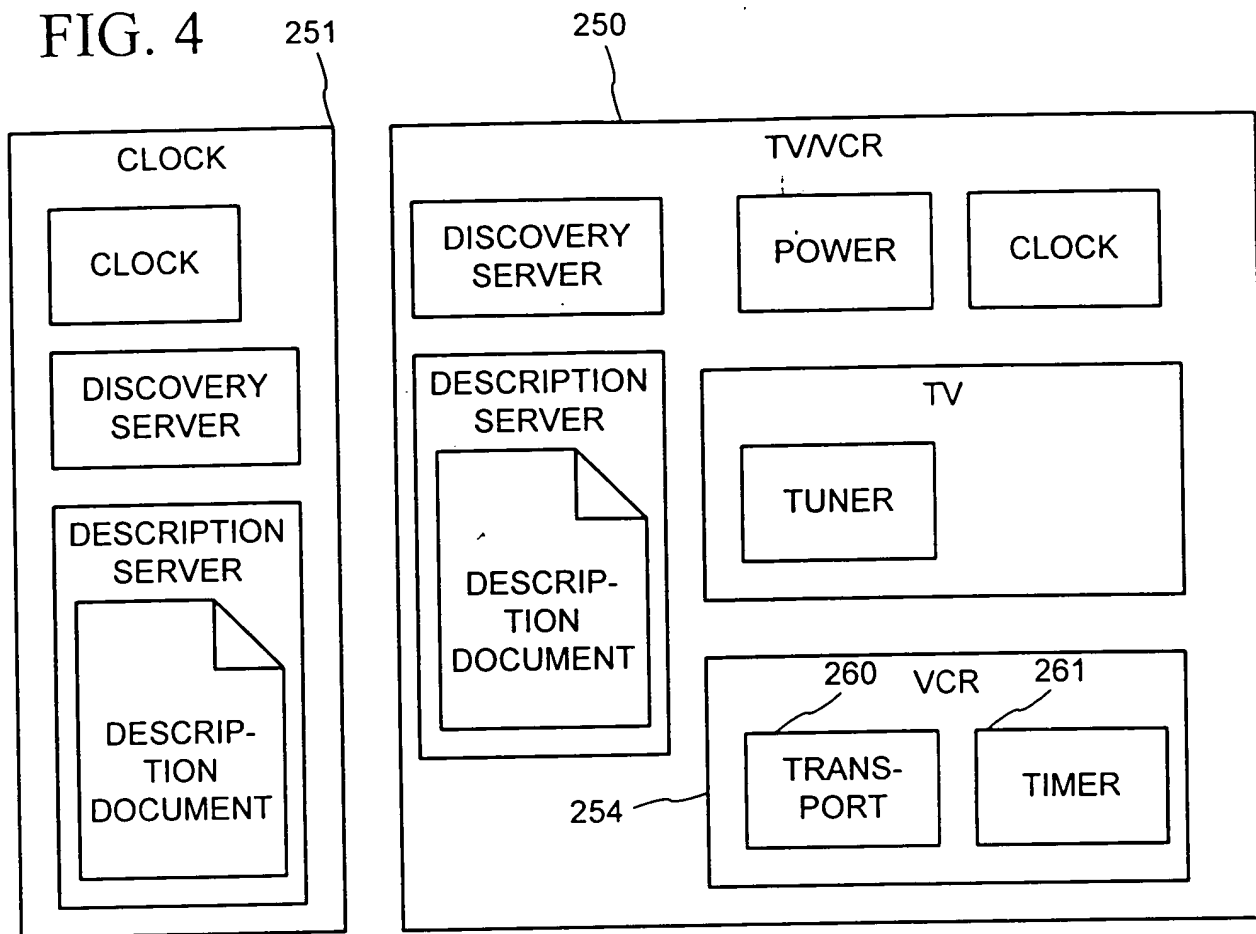


FIG. 5

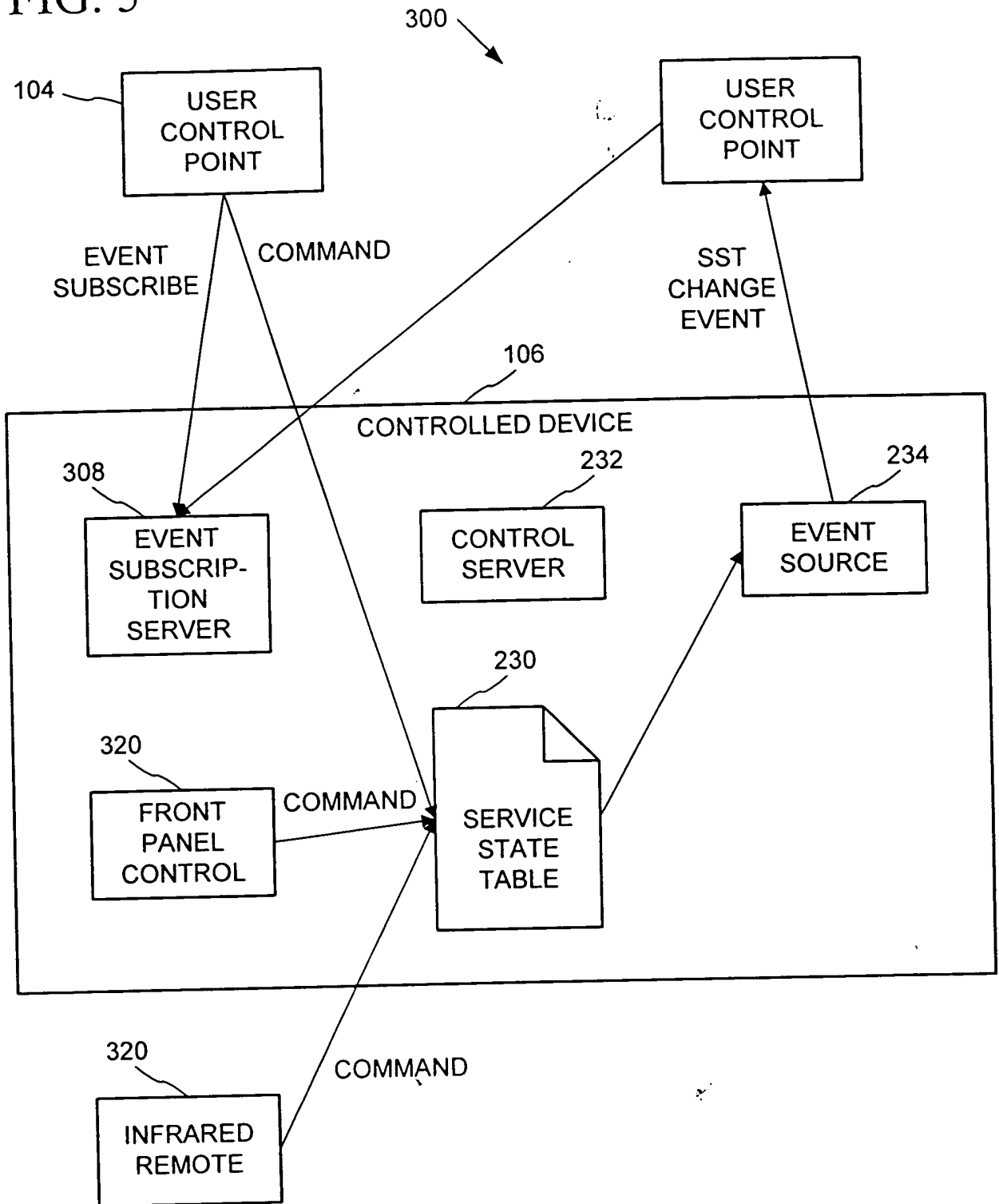
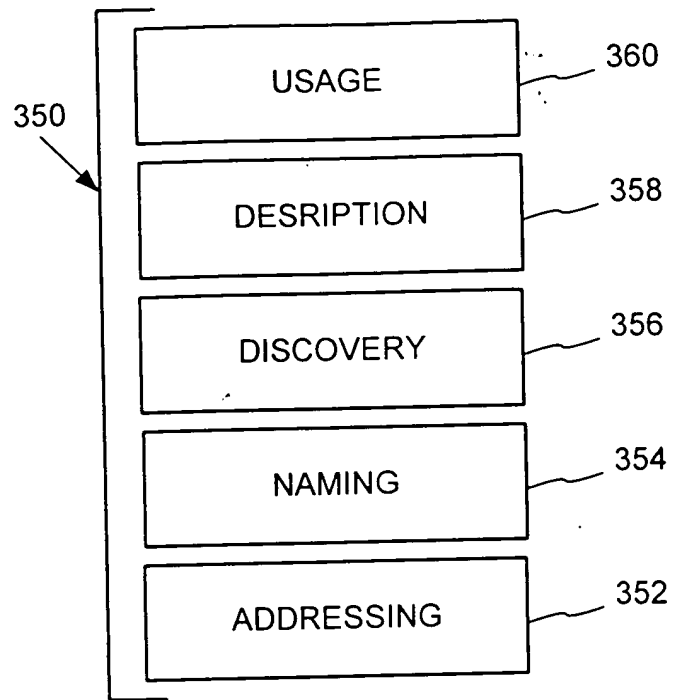
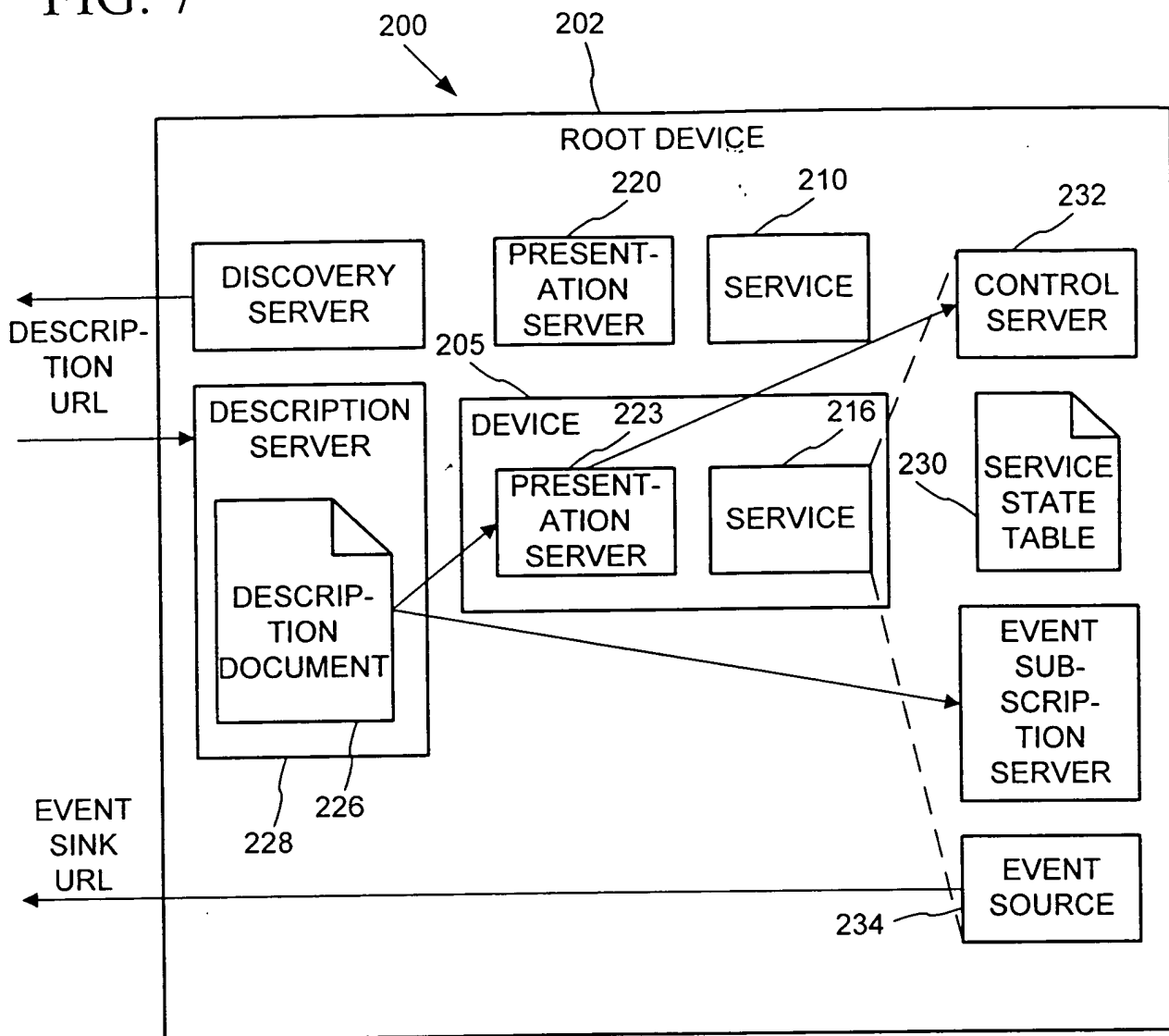


FIG. 6



001020-8TE96460

FIG. 7



000000-81E96460

FIG. 8

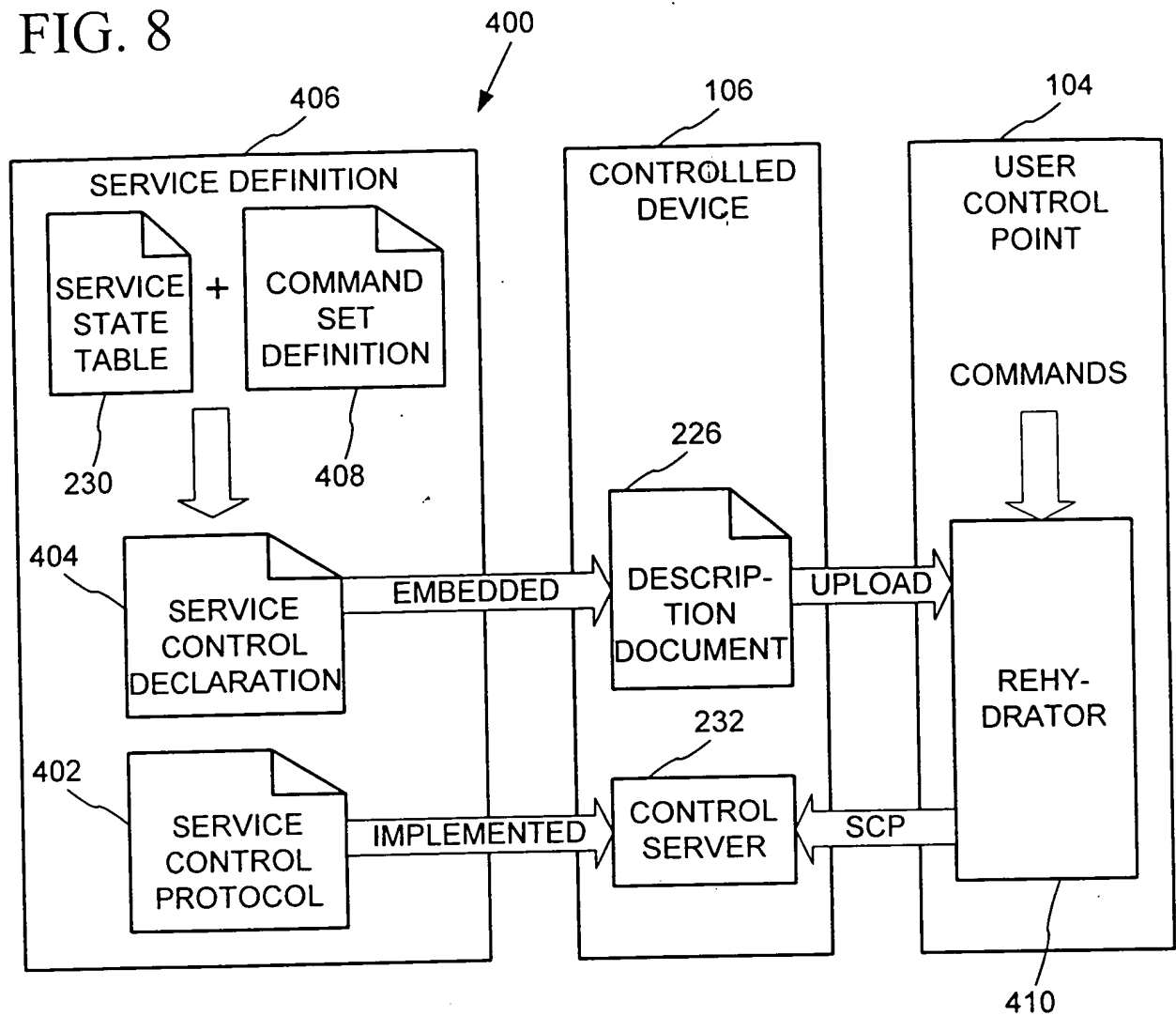
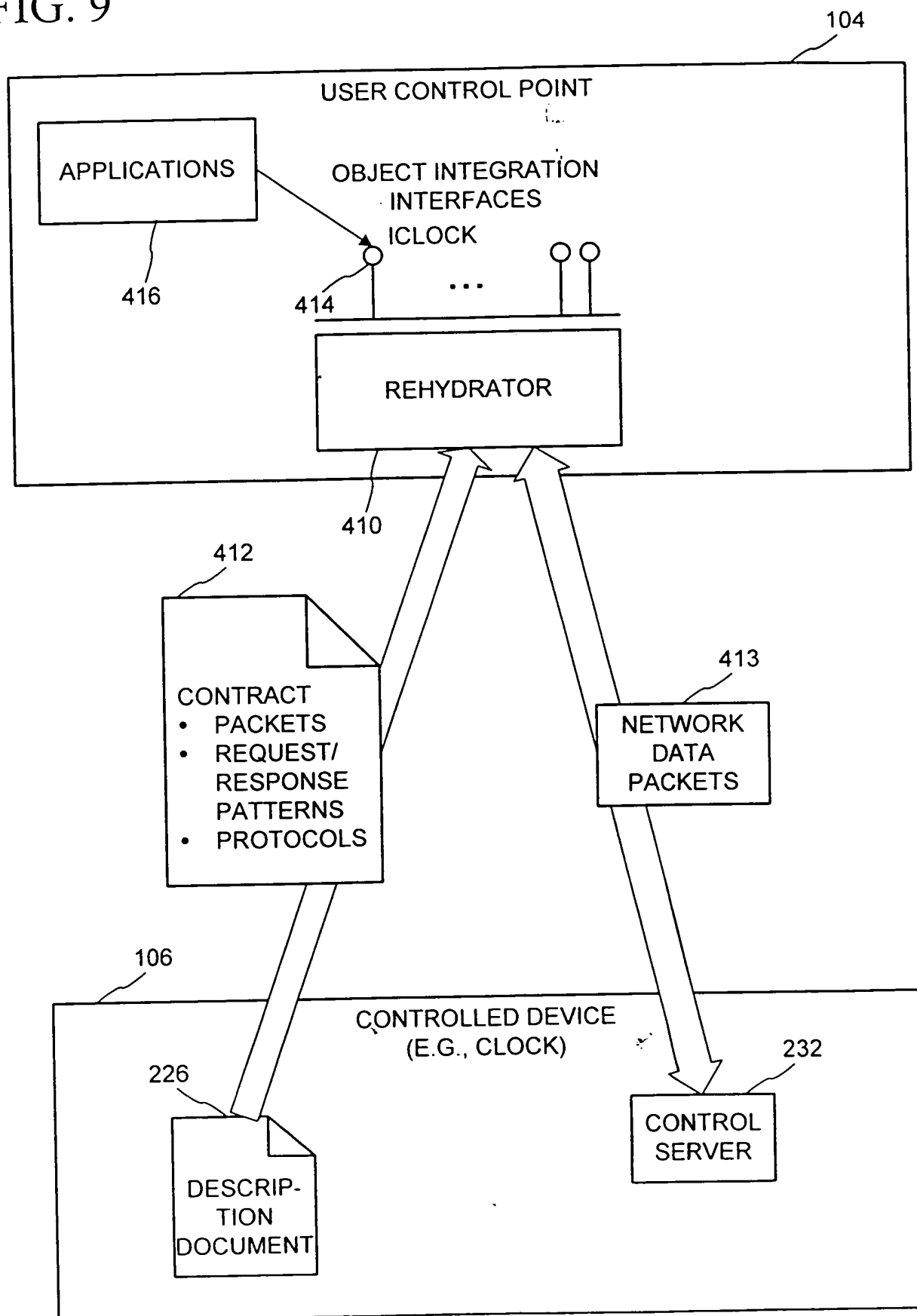


FIG. 9



000000-000000



FIG. 10

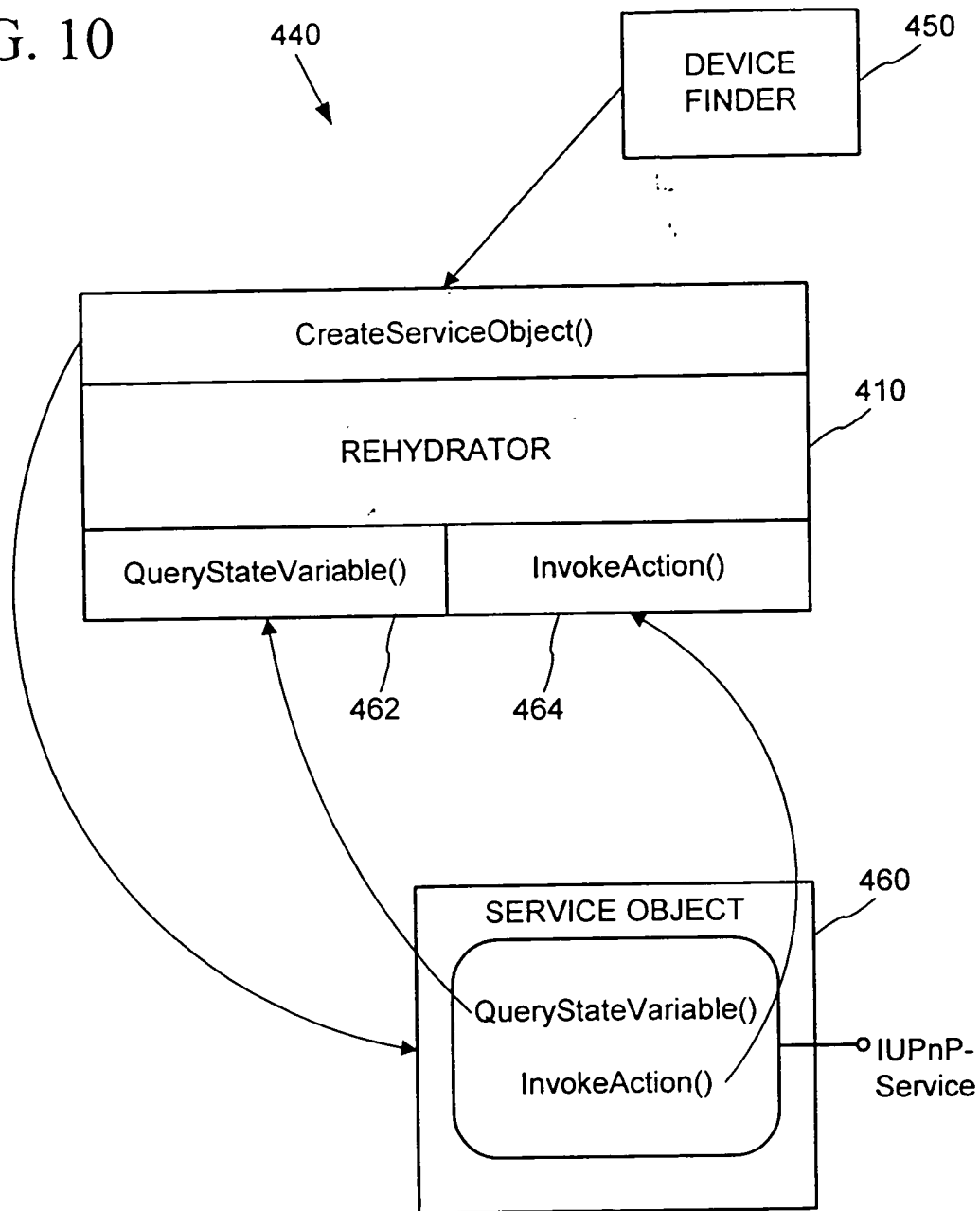


FIG. 11

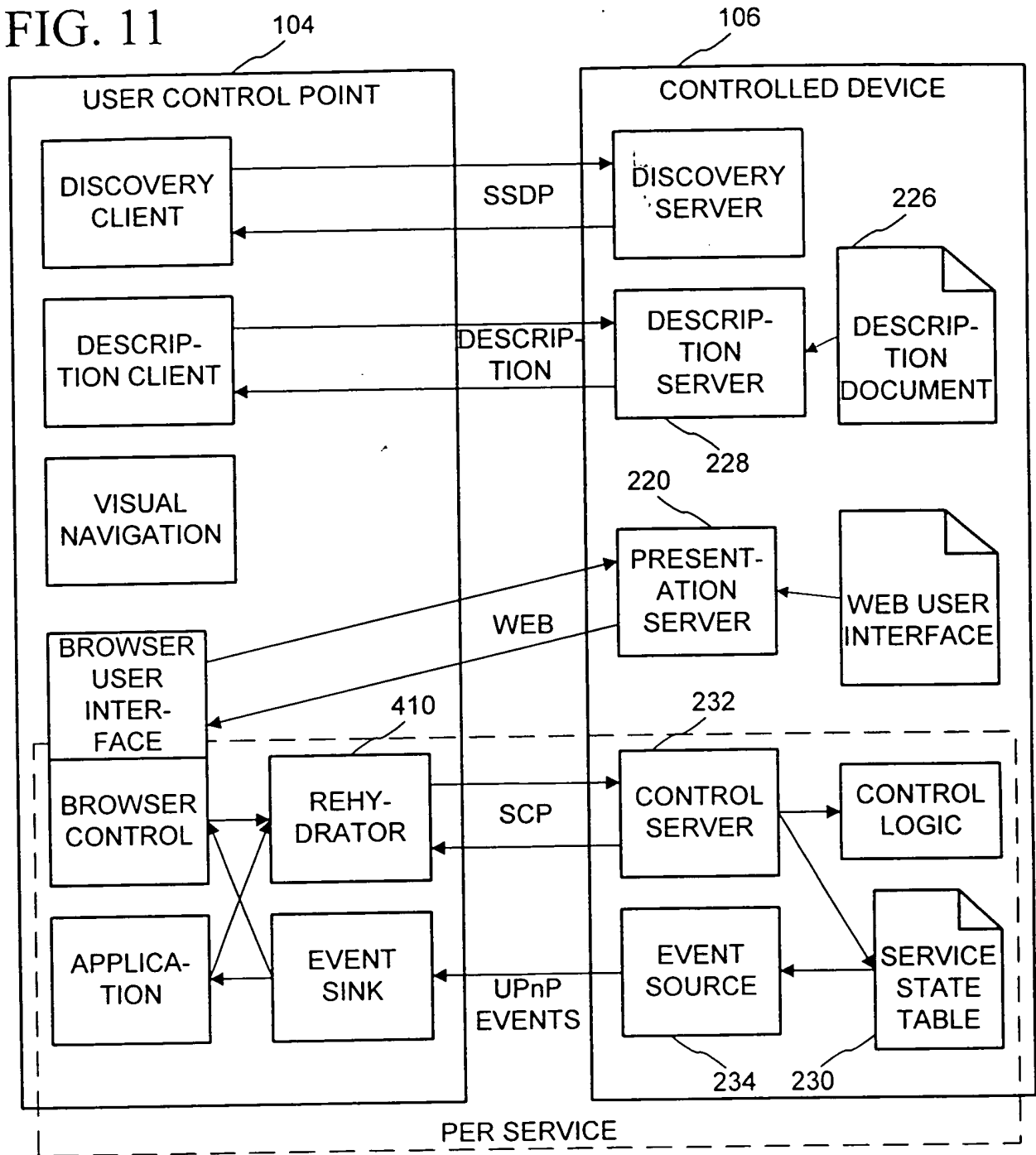
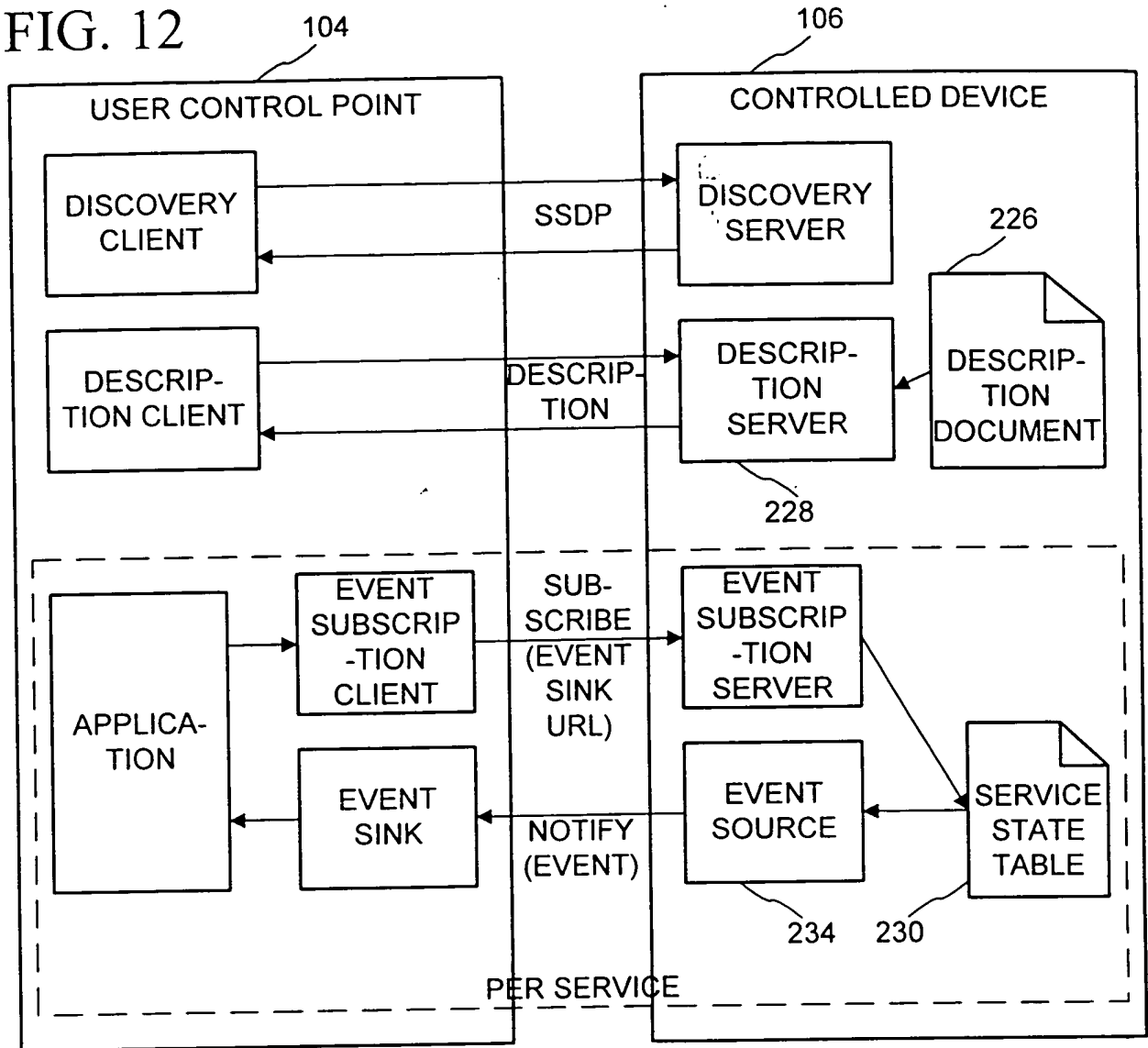


FIG. 12



001020-81E96460

FIG. 13

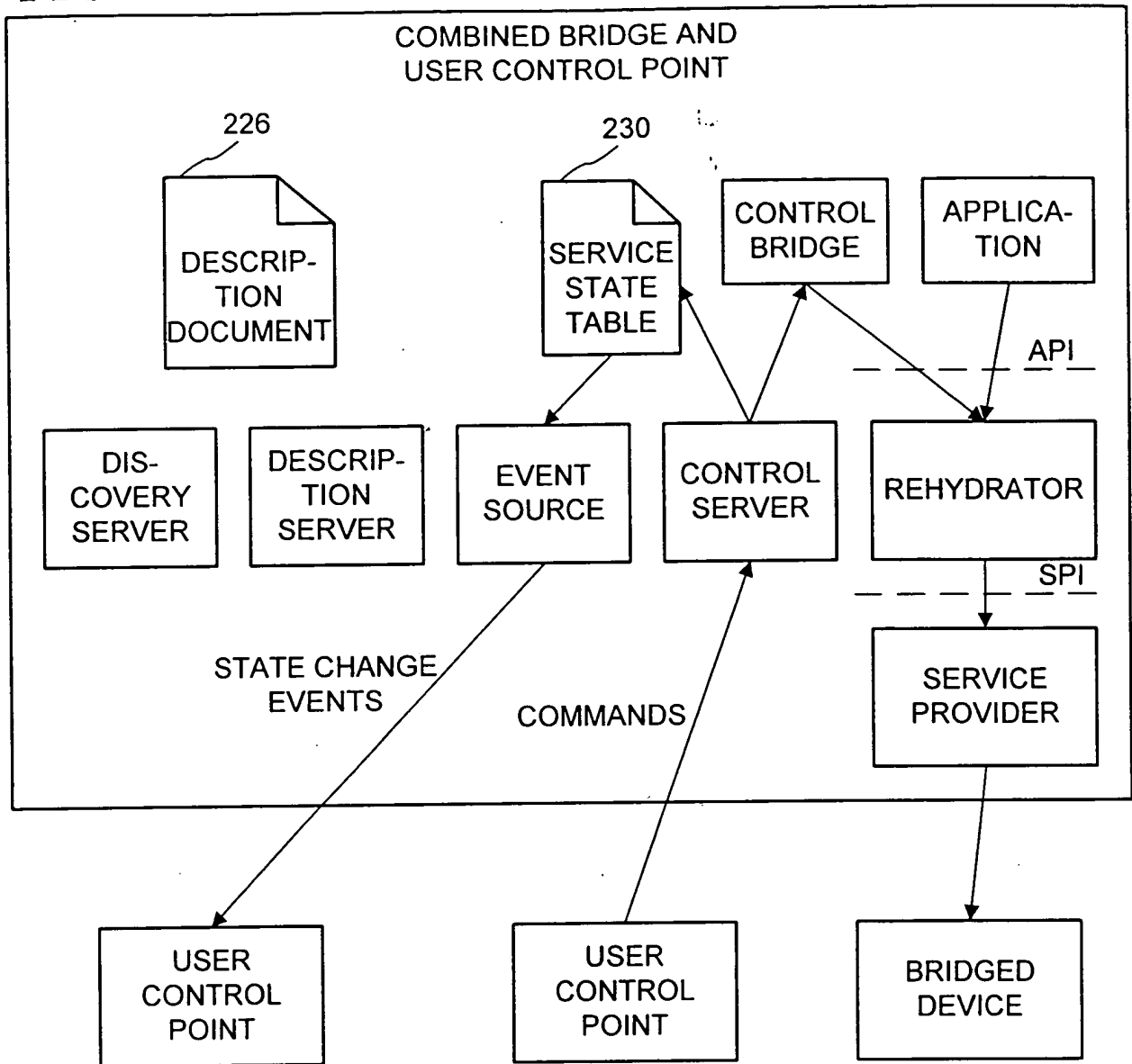
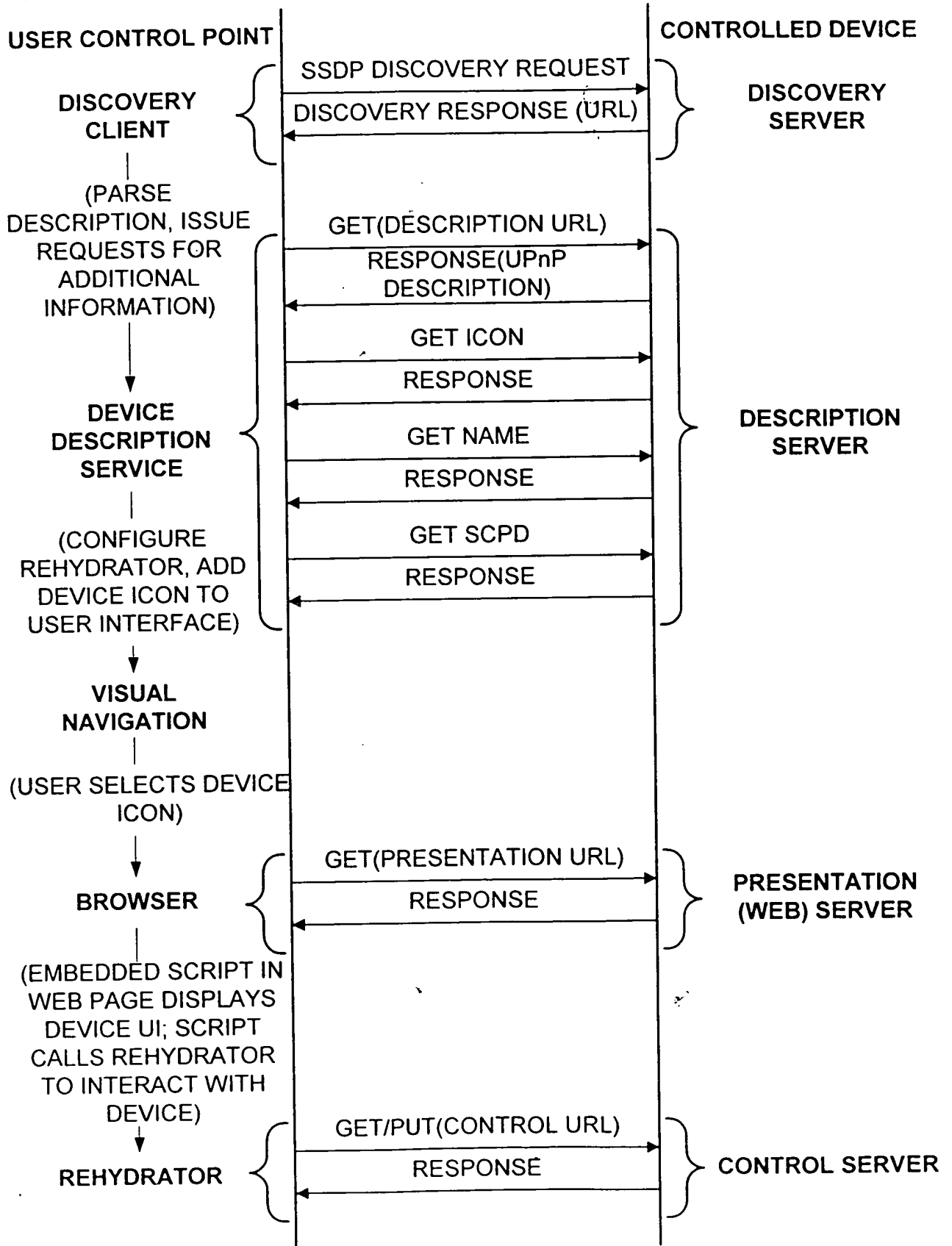


FIG. 14



001020-37E96460

FIG. 15

root

specVersionMajor  
specVersionMinor  
URLBase  
manufacturer  
manufacturerURL  
modelName  
modelNumber  
modelDescription  
modelURL  
UPC  
serialNumber  
device  
    UDN  
    friendlyName  
deviceType  
presentationURL  
iconList  
    icon  
        size  
        color  
        depth  
        imageType  
        imageURL  
    icon  
    icon  
service  
    serviceType  
    controlURL  
    eventSubURL  
    SCPD  
service  
service  
device  
    service  
    service  
    device  
        service  
device  
device

00000000000000000000000000000000







# FIG. 18

```

<contract>
  <protocol id="protocolDef">
    <HTTP version="1.1">
      <URL></URL>
      <M-POST>
        <MAN>http://www.microsoft.com/protocols/ext/XOAP</MAN>
      </M-POST>
      <HEADER name="Content-Type" value="text/xml" />
      <!-- Need to put in extension headers here -->
    </HTTP>
  </protocol>

  <RequestResponse name="queryStateVariable">
    <protocol is="protocolDef">
      <in is="queryStateVariable">
        <out is="queryStateVariableResponse">
          <error is="queryStateVariableResponse">
        </RequestResponse>

    <RequestResponse name="invokeAction">
      <protocol is="protocolDef">
        <in is="SerializedStream">
          <out is="invokeActionResponse">
            <error is="invokeActionResponse">
          </RequestResponse>

    <Schema name="upnp_scpd"
      xmlns="urn:schemas-microsoft-com:xml-data"
      xmlns:dt="urn:schemas-microsoft-com:datatypes">

      <!-- Common -->

      <ElementType name="_return" content="textOnly" dt:type="string" />
      <ElementType name="_fault" content="textOnly" dt:type="string" />

      <!-- Query State Variable Call -->

      <ElementType name="variableName" content="textOnly" dt:type="string" />

      <ElementType name="queryStateVariable" content="eltOnly" model="closed">
        <element type="variableName" />
      </ElementType>

      <!-- Query State Variable Response -->

```

FIG. 19

```

...
<ElementType name="queryStateVariableResponse" content="eltOnly"
model="closed">
  <group order="one">
    <element type="_return">
    <element type="_fault">
  </group>
</ElementType>

<!-- Invoke Action Call -->

<AttributeType name="main" dt:type="idref" />
<AttributeType name="headers" dt:type="idref" />
<AttributeType name="id" dt:type="id" />

<ElementType name="sequenceNumber" content="textOnly" dt:type="int">
  <AttributeType name="dt" dt:type="string" dt:values="int" />

  <attribute type="dt" />
</ElementType>

<ElementType name="headers" content="eltOnly" model="closed"
  <attribute type="id" required="yes" />
  <element type="sequenceNumber" />
</ElementType>

<ElementType name="actionName" content="textOnly" dt:type="string" />
<ElementType name="actionArg" content="textOnly" dt:type="string" />

<ElementType name="invokeAction" content="eltOnly" model="closed">
  <attribute type="id" required="yes" />

  <element type="actionName">
  <element type="actionArg" minOccurs="0" maxOccurs="*">
</ElementType>
...

```

001020-87E95450

FIG. 20

```
...
<ElementType name="SerializedStream" content="eltOnly" model="closed">
  <attribute type="main" required="yes" />
  <attribute type="headers" required="yes" />

  <element type="headers">
  <element type="invokeAction">

</ElementType>

<!-- Invoke Action Response -->

<ElementType name="invokeActionResponse" content="eltOnly" model="closed">
  <group order="one">
    <element type="_return">
    <element type="_fault">
  </group>
</ElementType>
</Schema>
</contract>
```

FIG. 21

```
<?xml version="1.0"?>
<Schema name="upnp_scpdl"
  xmlns="urn:schemas-microsoft-com:xml-data"
  xmlns:dt="urn:schemas-microsoft-com:datatypes">

  <!-- Common Elements and Attributes -->

  <ElementType name="name" content="textOnly" dt:type="string" />

  <!-- Service State Table -->

  <ElementType name="minimum" content="textOnly" dt:type="number" />
  <ElementType name="maximum" content="textOnly" dt:type="number" />
  <ElementType name="step" content="textOnly" dt:type="number" />

  <ElementType name="allowedValueRange" content="eltOnly" model="closed">
    <element type="minimum" />
    <element type="maximum" />
    <element type="step" minOccurs="0" />
  </ElementType>

  <ElementType name="allowedValue" content="textOnly" />

  <ElementType name="allowedValueList" content="eltOnly" model="closed">
    <element type="allowedValue" minOccurs="1" maxOccurs="*" />
  </ElementType>

  <ElementType name="dataType" content="textOnly" dt:type="string" />

  <ElementType name="stateVariable" content="eltOnly" model="closed">
    <element type="name" />
    ...
```

# FIG. 22

```

...
    <element type="dataType" />

    <group minOccurs="0" maxOccurs="1" order="one">
        <element type="allowedValueRange" />
        <element type="allowedValueList" />
    </group>
</ElementType>

<ElementType name="deviceStateTable" content="eltOnly" model="closed">
    <element type="stateVariable" minOccurs="1" maxOccurs="*" />
</ElementType>

<!-- Action List -->

<ElementType name="relatedStateVariable" content="textOnly" dt:type="string" />

<ElementType name="argument" content="eltOnly" model="closed">
    <element type="name" />
    <element type="relatedStateVariable" />
</ElementType>

<ElementType name="action" content="eltOnly" model="closed">
    <element type="name" />
    <element type="argument" minOccurs="0" maxOccurs="*" />
</ElementType>

<ElementType name="actionList" content="eltOnly" model="closed">
    <element type="action" minOccurs="0" maxOccurs="*" />
</ElementType>

<!-- Root Element -->

<ElementType name="dcpd" content="eltOnly" model="closed">
    <element type="deviceStateTable" />
    <element type="actionList" />
</ElementType>
</Schema>

```

FIG. 23

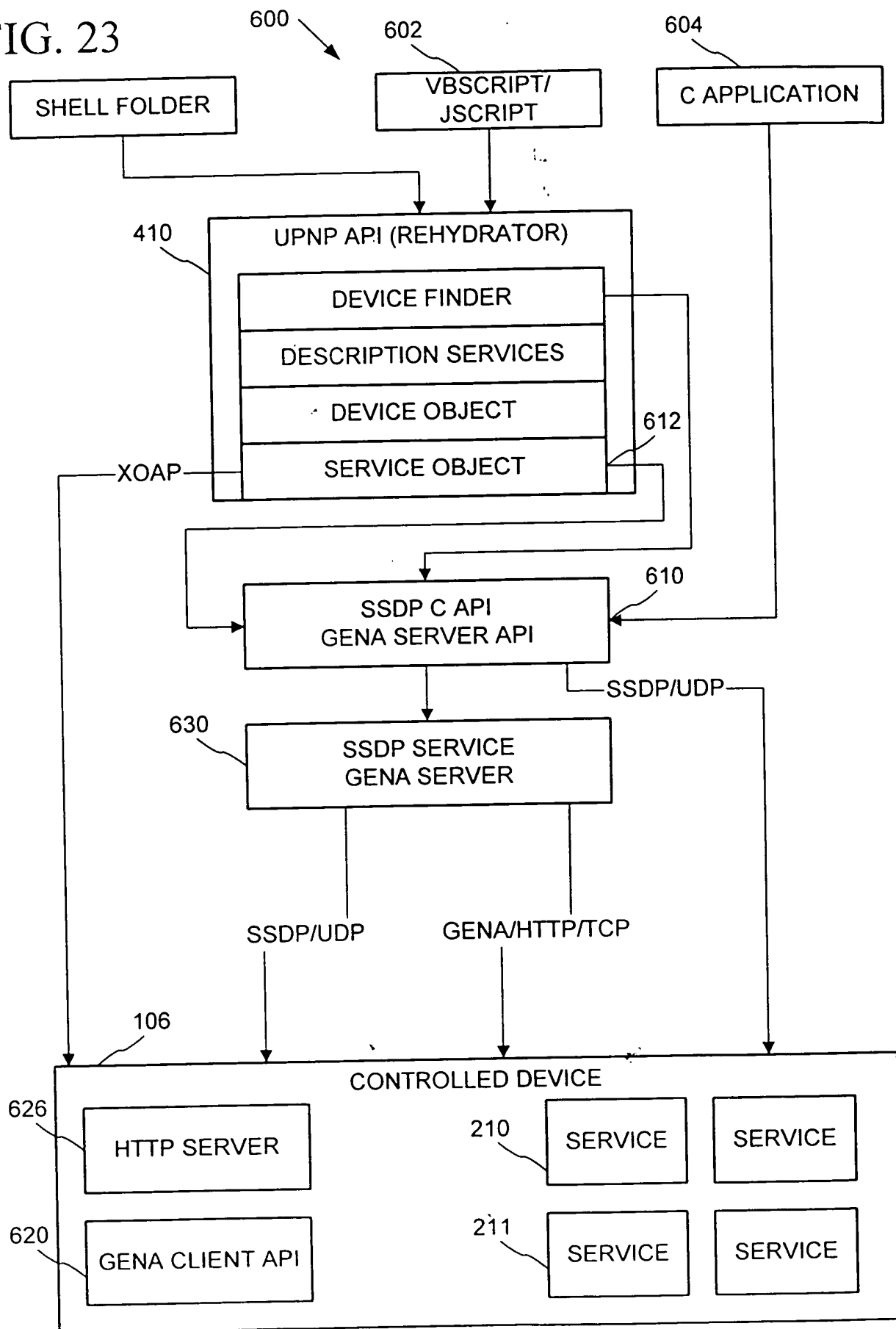
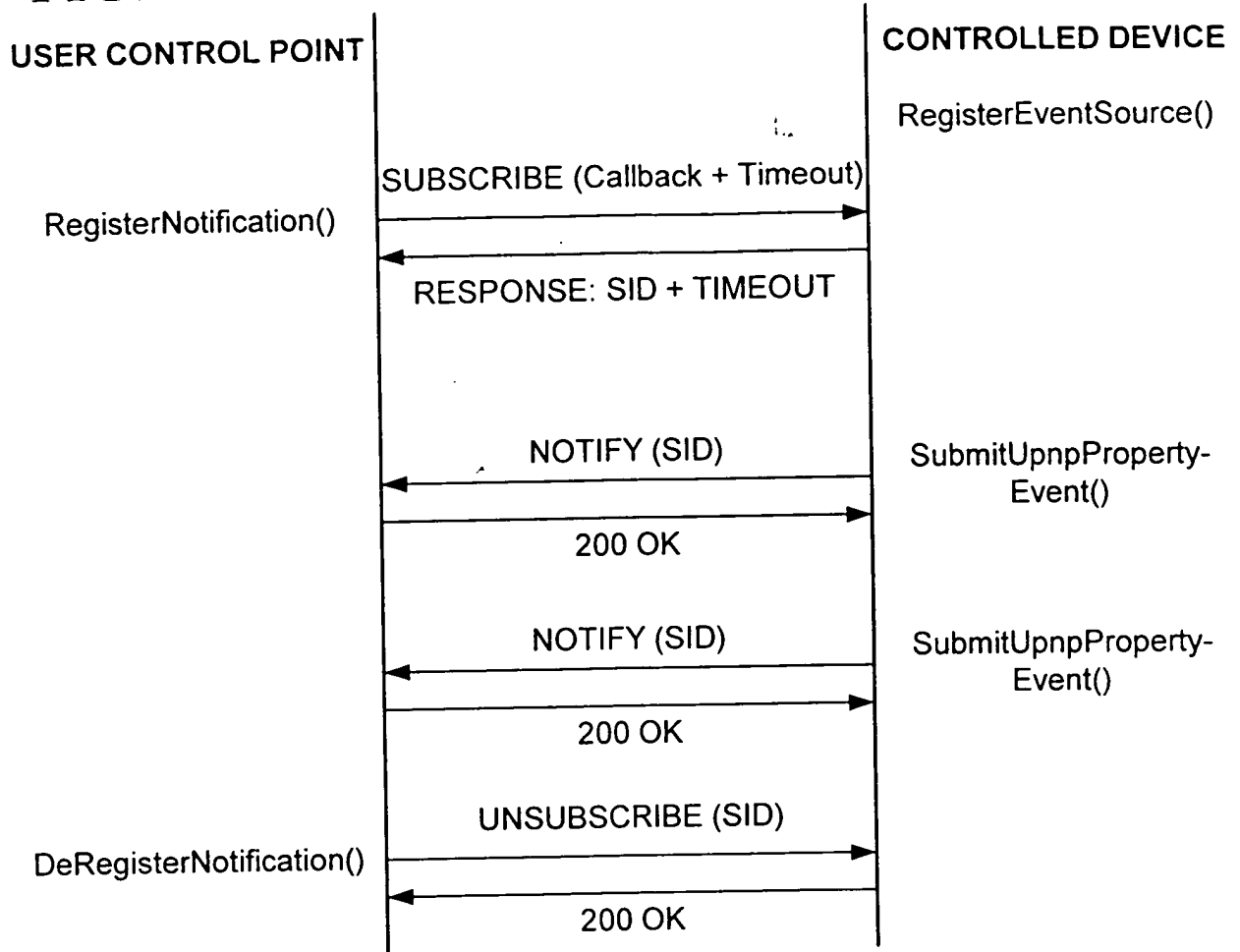


FIG. 24



001020" ETE 55450

FIG. 25

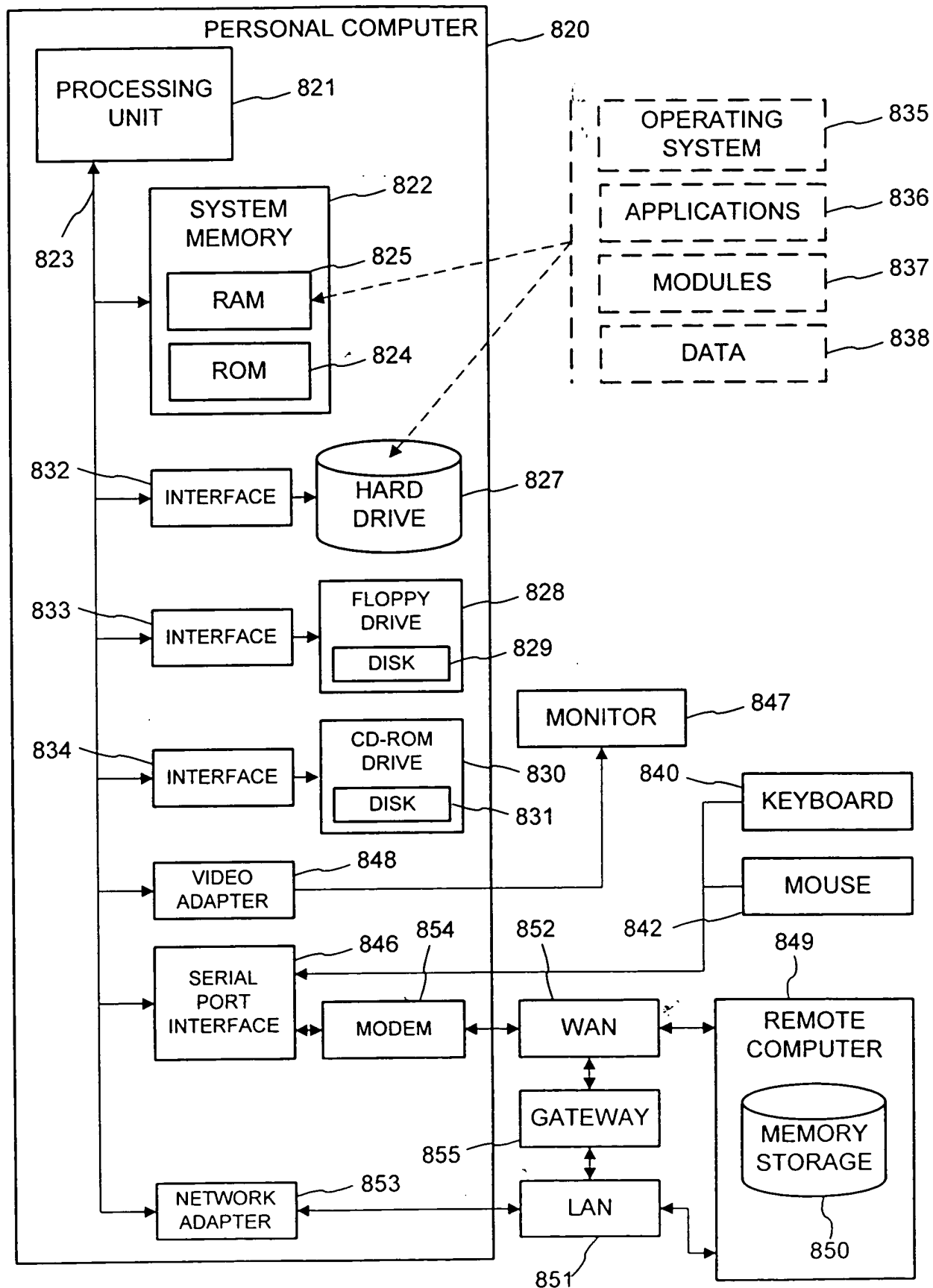




FIG. 26

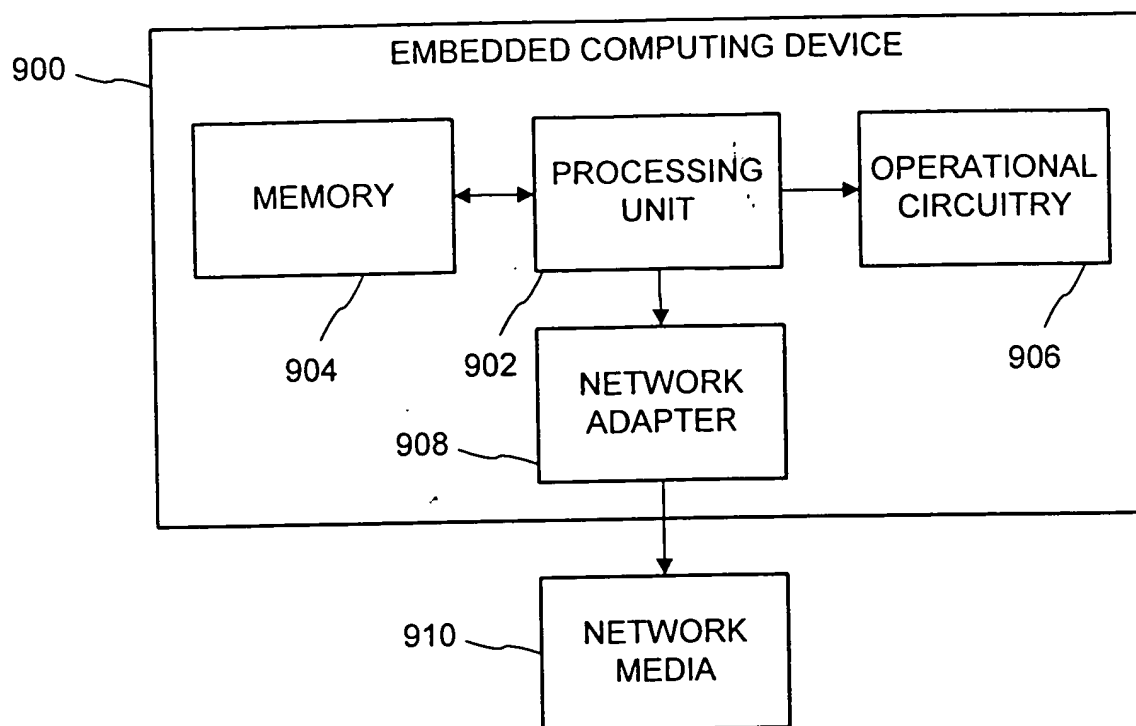
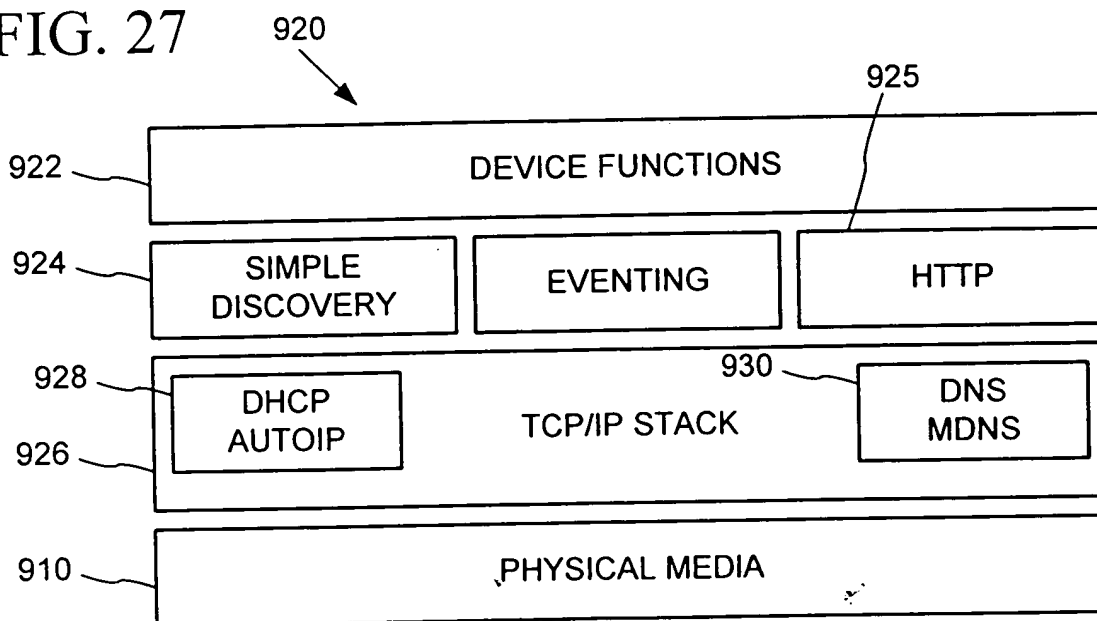
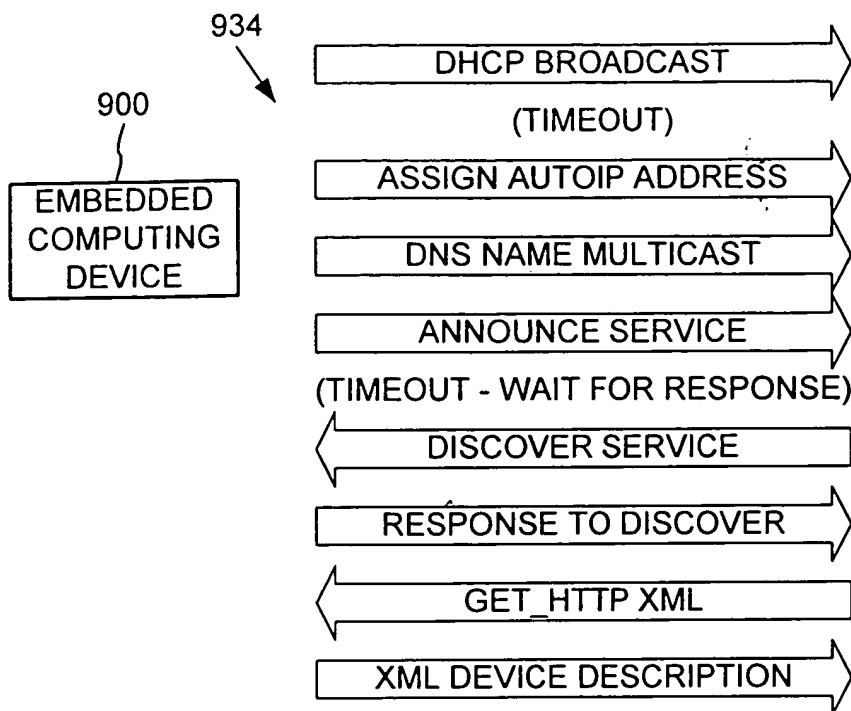


FIG. 27



# FIG. 28



# FIG. 29

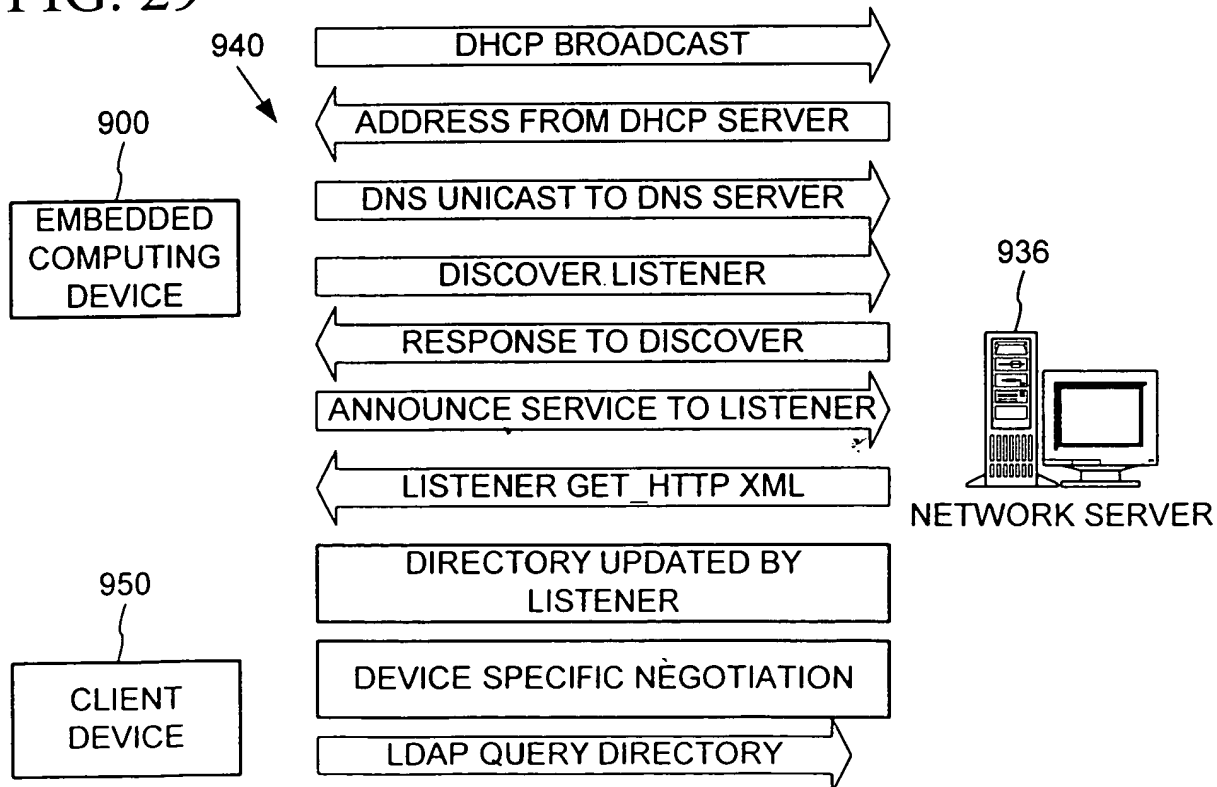
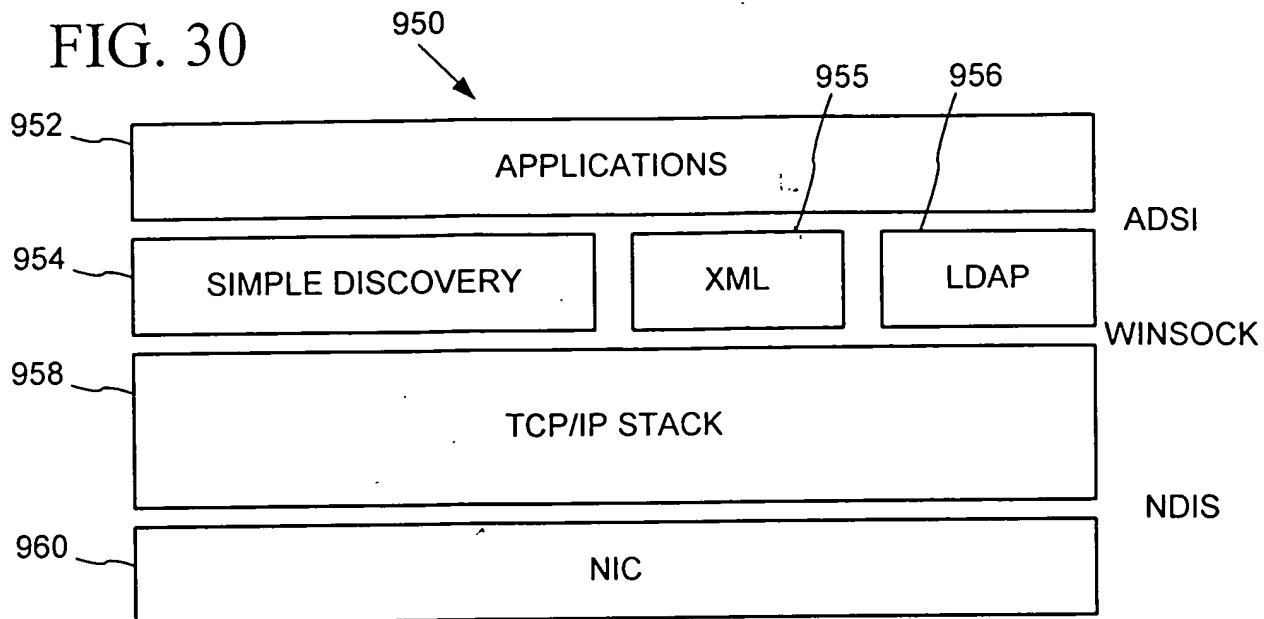


FIG. 30



004020-31E96460

FIG. 31

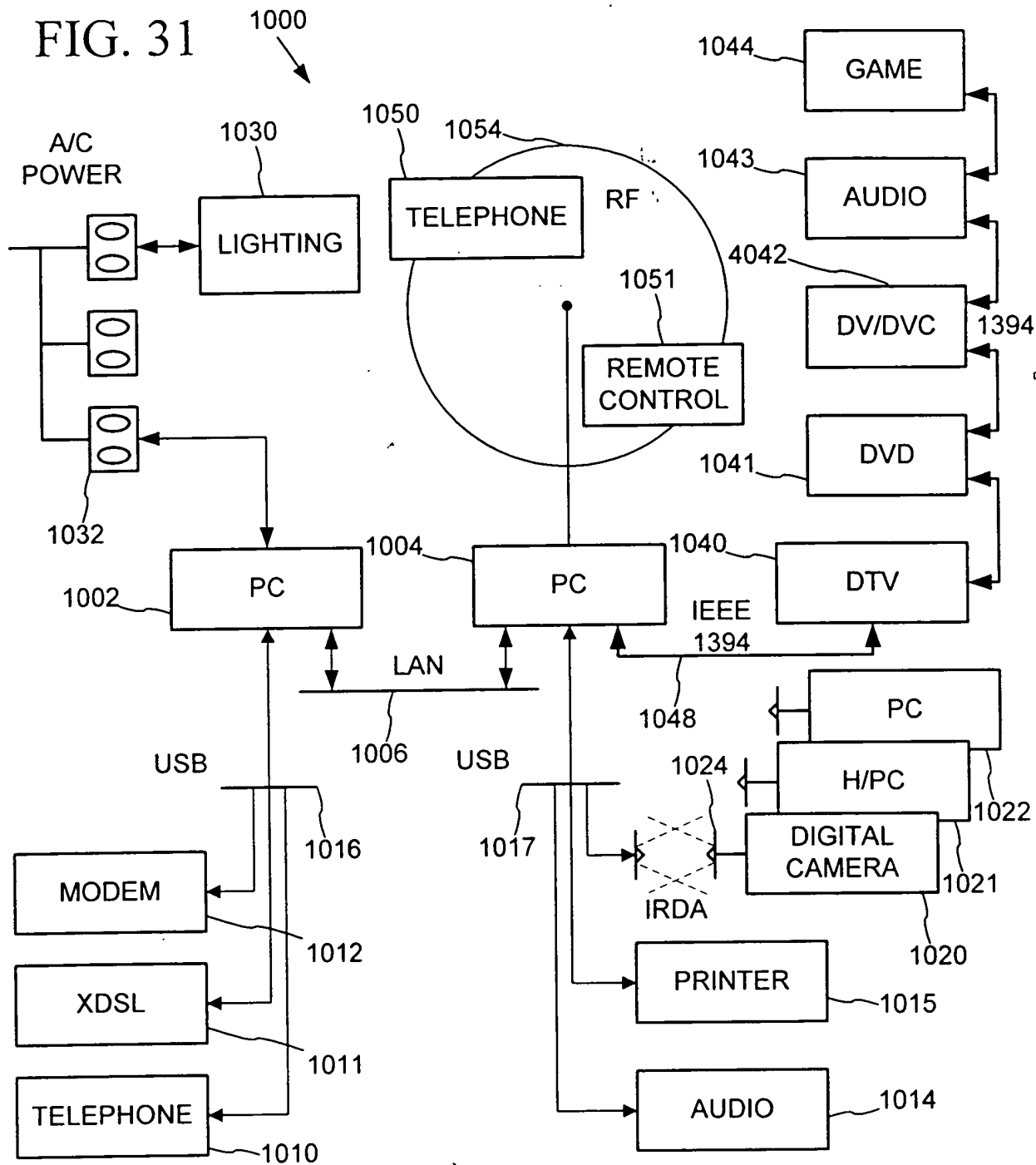


FIG. 31



FIG. 33

001020-8796460

...  
parameters: ppudDeviceRoot, On return, this refers to the "root" device of the current device tree. The root device is the topmost parent of the current device. If the current device is the root device this method will set \*ppudDeviceRoot to null, and return S\_FALSE.

return values: S\_OK, \*ppudDeviceRoot contains a reference to the root device. S\_FALSE, the current device is the root device. \*ppudDeviceRoot is null.

[propget, id(DISPID\_UPNPDEVICE\_PARENT),  
helpstring("returns the parent of the current device")]  
HRESULT ParentDevice([out, retval] IUPNPDevice \*\* ppudDeviceParent);

parameters: ppudDeviceParent, On return, if the device has a parent, this is the address of a IUPNPDevice object which can describe the parent. This must be released when no longer needed. If the device has no parent (it is a "root" device), then this value will be set to null.

return values: S\_OK, ppudDeviceParent contains a reference to the device's parent. S\_FALSE, the current device is the root device, which has no parent. \*ppudDeviceRoot is null.

[propget, id(DISPID\_UPNPDEVICE\_CHILDREN),  
helpstring("returns a collection of the children of the current device")]  
HRESULT Children([out, retval] IUPNPDevices \*\* ppudChildren);

parameters: ppudChildren, On return, this is the address of a newly-created IUPNPDevices collection that can enumerate this device's children. This must be released when no longer needed. If the device has no children, this method will return a collection object with a length of zero.

return values: S\_OK, ppudChildren contains a list of the device's children.

[propget, id(DISPID\_UPNPDEVICE\_UDN),  
helpstring("returns the UDN of the device")]  
HRESULT UniqueDeviceName([out, retval] BSTR \* pbstrUDN);

parameters: pbstrUDN, On return, this contains the address of a newly-allocated string which contains the device's Unique Device Name (UDN). The UDN is globally unique across all devices - no two devices will ever have the same UDN. This value must be freed when no longer needed.

return values: S\_OK pbstrUDN contains the UDN of the device  
...



FIG. 35

...

helpstring("returns the device type URI")]  
HRESULT Type([out, retval] BSTR \* pbstrType);  
parameters: pbstrType, On return, this contains the address of a newly-allocated string containing the device's type URI. This value must be freed when no longer needed.  
return values: S\_OK, bstrType contains the type URI of the device, and must be freed when no longer needed.

[propget, id(DISPID\_UPNPDEVICE\_SERVICES),  
helpstring("returns the collection of services exposed by the device")]  
HRESULT Services([out, retval] IUPNPServices \*\* ppusServices);  
parameters: ppusServices, On return, this is the address of a newly-created IUPNPServices collection that can enumerate the services exposed by the device. This must be released when no longer needed. If the device exposes no services, this method will return a collection object with a length of zero.  
return values: S\_OK, pusServices contains a list of the device's children.

[propget, id(DISPID\_UPNPDEVICE\_SERVICEIDENTIFIER),  
helpstring("returns the (optional) service identifier of the device")]  
HRESULT ServiceIdentifier([out, retval] BSTR \* pbstrServiceID );  
parameters: pbstrServiceID, On return, this contains the address of a newly-allocated string containing the contents of the device's ServiceIdentifier element, if the device specifies one. This value must be freed when no longer needed. If the device does not specify a ServiceIdentifier value, this parameter will be set to null.  
return value: S\_OK, bstrServiceID contains the service identifier of the device. pbstrServiceID must be freed. S\_FALSE, the device did not specify a service identifier. \*pbstrServiceID is set to null.  
note having a ServiceIdentifier is mutually exclusive with having services. Any device will either have a list of services or a ServiceIdentifier, but not both.

[id(DISPID\_UPNPDEVICEDESCRIPTION\_LOADSMALLICON),  
helpstring("loads a small (titlebar-sized) icon representing the device, encoded in the specified format")]  
HRESULT LoadSmallIcon([in] BSTR bstrEncodingFormat,  
[out, retval] BSTR \* pbstrIconURL);  
parameters:

...





[illegible]

note: if the object does not provide any description information, an empty property bag will be returned. See [SetPhysicalLocation](#) for a listing of defined values in a physical location property bag.

**[id(DISPID\_UPNPDEVIDESDESCRIPTION\_SETPHYSICALLOCATION),**  
**helpstring("writes a set of properties describing the device's physical location to**  
**the device")]**  
**HRESULT SetPhysicalLocation([in] IUPNPPropertyBag \* pupl);**  
**parameters:** pupl    A UPNPPropertyBag object which contains the name-  
value pairs representing the device's current location. the function will not free the  
object.  
**return values:** S\_OK            he device has been updated with the supplied  
physical location information

note: the following are standard values in the physical location property bag: country, campus, building, floor, wing, room, latitude, longitude, altitude. These values can be used programmatically to implement sorting or filtering functionality based on the device's location. Additionally the property bag supports the following value: description, which contains a user-displayable string representing a device's location which does not have programmatic significance. Additionally, the physical location update will be made on the device alone, and will not be reflected in the current device object. After a successful call to this method, PhysicalLocation will continue to return the 'old' value. To read the device's current name, the caller must re-load the device's description.

...

# FIG. 38

```

...
[propget, id(DISPID_UPNPDEVICEDESCRIPTION_PRODUCTNAME),
    helpstring("a displayable string containing the product name")]
HRESULT ProductName([out, retval] BSTR * pbstr);
    parameters: pbstr on return, the address of a newly-allocated string
containing the product name of the device.
    return values: S_OK pbstr contains a newly-allocated string that must
be freed when no longer needed.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_DESCRIPTION),
    helpstring("displayable summary of the device's function")]
HRESULT Description([out, retval] BSTR * pbstr);
    parameters: pbstr on return, the address of a newly-allocated string
containing a short description of the device meaningful to the user.
    return values: S_OK pbstr contains a newly-allocated string that must
be freed when no longer needed.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_MODELNAME),
    helpstring("displayable model name")]
HRESULT ModelName([out, retval] BSTR * pbstr);
    parameters: pbstr on return, the address of a newly-allocated string
containing the manufacturer's model name of the device.
    return values: S_OK pbstr contains a newly-allocated string that must
be freed when no longer needed.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_SERIALNUMBER),
    helpstring("displayable serial number")]
HRESULT SerialNumber([out, retval] BSTR * pbstr);
    parameters: pbstr on return, the address of a newly-allocated string
containing the manufacturer's serial number of the device.
    return values: S_OK pbstr contains a newly-allocated string that must
be freed when no longer needed.
    note: a device's serial number is not guaranteed to be globally unique. The
DeviceUniqueName should always be used to distinguish devices.

[propget, id(DISPID_UPNPDEVICEDESCRIPTION_MANUFACTURERNAME),
    helpstring("displayable manufacturer name")]
HRESULT ManufacturerName([out, retval] BSTR * pbstr);
    parameters
...

```

FIG. 39

...  
pbstr, on return, the address of a newly-allocated string containing the name of the device's manufacturer.

return values: S\_OK, pbstr contains a newly-allocated string that must be freed when no longer needed.

[propget, id(DISPID\_UPNPDEVICEDESCRIPTION\_MANUFACTURERURL),  
helpstring("URL to the manufacturer's website")]

HRESULT ManufacturerURL([out, retval] BSTR \* pbstr);

parameters: pbstr, on return, the address of a newly-allocated string containing the URL of the manufacturer's website.

return values: S\_OK, pbstr contains a newly-allocated string that must be freed when no longer needed.

[propget, id(DISPID\_UPNPDEVICEDESCRIPTION\_MODELNAME),  
helpstring("displayable model name")]

HRESULT ModelName([out, retval] BSTR \* pbstr);

parameters: pbstr, on return, the address of a newly-allocated string containing the manufacturer's model name for the device.

return values: S\_OK, pbstr contains a newly-allocated string that must be freed when no longer needed.

[propget, id(DISPID\_UPNPDEVICEDESCRIPTION\_SUPPORTLIST),  
helpstring("technical support contact information")]

HRESULT SupportList([out, retval] BSTR \* pbstr);

parameters: pbstr, on return, the address of a newly-allocated, multi-line string containing phone numbers and other information that can guide the user to technical support. This string must be freed when no longer needed.

return values: S\_OK, pbstr contains a newly-allocated string that must be freed when no longer needed.

[propget, id(DISPID\_UPNPDEVICEDESCRIPTION\_FAQLIST),  
helpstring("FAQ access display information")]

HRESULT FAQList([out, retval] BSTR \* pbstr);

parameters: pbstr, on return, the address of a newly-allocated, multi-line string containing FAQ information that can provide the user with URLs at which device FAQs may be located.

return values: S\_OK, pbstr contains a newly-allocated string that must be freed when no longer needed.

...

FIG. 40

```
...  
[propget, id(DISPID_UPNPDEVICEDESCRIPTION_UPDATELIST),  
  helpstring("information explaining where the user can update the device's  
firmware")]  
HRESULT UpdateList([out, retval] BSTR * pbstr);  
  parameters: pbstr, on return, the address of a newly-allocated, multi-line  
string containing information and URLs from which the user can download updates  
for the device's firmware.  
  return values: S_OK, pbstr contains a newly-allocated string that must be  
freed when no longer needed.
```

# FIG. 41

```
[
    object,
    uuid(FDBC0C73-BDA3-4C66-AC4F-F2D96FDAD68C),
    dual,
    helpstring("IUPNPDevices Interface"),
    pointer_default(unique)
]
IUPNPPropertyBag
{

    [propget, id(DISPID_UPNP_PROPERTYBAG_READ),
     helpstring("reads a value from the property bag")]
    HRESULT Read([in] BSTR bstrName, [out, retval] VARIANT * pvarResult);
        parameters: bstrName, name of the property to read. case is ignored.
        pvarResult value of the property. if the property does not exist, this is of type
        VT_EMPTY
        return values: S_OK, the value was found in the property bag, and returned
        in pvarResult. S_FALSE, there was no value with the given name in the property
        bag. *pvarResult is of type VT_EMPTY

    [propget, id(DISPID_UPNP_PROPERTYBAG_WRITE),
     helpstring("writes a value to the property bag")]
    HRESULT Write([in] BSTR bstrName, [in] VARIANT * pvarValue);
        parameters: bstrName, name of the property to write. case is preserved
        when writing. The supplied value will replace any other values of the same name,
        even if they differ in case. pvarValue, value of the property to write.
        return values: S_OK, the value was written to the property bag, replacing the
        value currently associated with this property, if it existed.

    [propget, id(DISPID_UPNP_PROPERTYBAG_DELETE),
     helpstring("removes a value from the property bag")]
    HRESULT Delete([in] BSTR bstrName);
        parameters: bstrName, name of the value to remove from the property bag.
        case is ignored when finding a value to remove.
        return values: S_OK, the value has been removed from the property bag.
        S_FALSE, the value was not found in the property bag.

};
```

FIG. 42

```
[
  object,
  uuid(A295019C-DC65-47DD-90DC-7FE918A1AB44),
  dual,
  helpstring("IUPNPSERVICE Interface"),
  pointer_default(unique)
]
interface IUPNPSERVICE : IDispatch
{
  [id(1), helpstring("method GetProperty")]
  HRESULT GetProperty(
    [in] BSTR bstrPropertyName,
    [out, retval] VARIANT *pValue
  );

  [id(2), helpstring("method InvokeAction")]
  HRESULT InvokeAction(
    [in] BSTR bstrActionName,
    [in] VARIANT saActionArgs,
    [out, retval] long *plStatus
  );

  [propget, id(3), helpstring("property DCPI")]
  HRESULT DCPI(
    [out, retval] BSTR *pVal
  );

  [propget, id(4),
    helpstring("returns a manufacturer-defined extension property")]
  HRESULT VendorExtension([out, retval] VARIANT * pvarValue );
    parameters: pvarValueOn return, this variant is filled with the value of the
    "extension" element. If none exists, pvarValue is set to VT_EMPTY
    return values: S_OK, varValue is set to the extension element. S_FALSE,
    no vendor extension element exists. pvarValue is VT_EMPTY
```

00000000-0000-0000-0000-00000000

FIG. 43

```
[
  object,
  uuid(FDBC0C73-BDA3-4C66-AC4F-F2D96FDAD68C),
  dual,
  helpstring("IUPNPDevices Interface"),
  pointer_default(unique)
]
  interface IUPNPDevices : IDispatch
  {
    [propget, id(1), helpstring("property Count")]
    HRESULT Count(
      [out, retval] long *pVal
    );

    [propget, id(DISPID_NEWENUM), helpstring("property _NewEnum")]
    HRESULT _NewEnum(
      [out, retval] LPUNKNOWN *pVal
    );

    [propget, id(DISPID_VALUE), helpstring("property Item")]
    HRESULT Item(
      [in] long lIndex,
      [out, retval] VARIANT *pVal
    );
  };
}
```

00000000-0000-0000-0000-00000000



FIG. 44

```
[
  object,
  uuid(3F8C8E9E-9A7A-4DC8-BC41-FF31FA374956),
  dual,
  helpstring("IUPNPServices Interface"),
  pointer_default(unique)
]
interface IUPNPServices : IDispatch
{
  [propget, id(1), helpstring("property Count")]
  HRESULT Count(
    [out, retval] long *pVal
  );

  [propget, id(DISPID_NEWENUM), helpstring("property _NewEnum")]
  HRESULT _NewEnum(
    [out, retval] LPUNKNOWN *pVal
  );

  [propget, id(DISPID_VALUE), helpstring("property Item")]
  HRESULT Item(
    [in] long lIndex,
    [out, retval] VARIANT *pVal
  );
};
```

001020-87E96460

FIG. 45

```

<contract>

  <protocol id="protocolDef">
    <HTTP version="1.1">
      <URL> http://investor.msn.com/stockquote </URL>
      <M-POST>
        <MAN> http://www.upnp.org/service-control/m-post </MAN>
      <M-POST>
        <HEADER name="Content-Type" value="text/xml" />
      </HTTP>
    </protocol>

    <RequestResponse name="getQuote">
      <protocol is="protocolDef" />
      <in is="symbol" />
      <out is="stockQuote" />
      <error is="error" />
    </RequestResponse>

    <RequestResponse name="getQuotes">
      <protocol is="protocolDef" />
      <in is="symbols" />
      <out is="stockQuotes" />
      <error is="error" />
    </RequestResponse>

    <!-- // schema definition follows -->

    <schema xmlns="urn:schema-microsoft-com:xml-data"
      xmlns:dt="urn:schema-microsoft-com:datatypes">

      <ElementType name="symbol" dt:type="string" />

      <ElementType name="symbols">
        <element type="symbol" maxOccurs="*" />
      </ElementType>

      <ElementType name="stockQuote">
        <element type="company" />
        <element type="ticker" />

```

001020-31E96460

FIG. 46

```

...
    <element type="previousClose" />
    <element type="openingTrade" />
    <element type="lastTrade" />
    <element type="volume" />
  </ElementType>

  <ElementType dt:type="string" name="company" />
  <ElementType dt:type="string" name="ticker" />
  <ElementType dt:type="string" name="previousClose" />
  <ElementType dt:type="string" name="openingTrade" />
  <ElementType dt:type="string" name="lastTrade" />
  <ElementType dt:type="string" name="volume" />

  <ElementType name="stockQuotes">
    <element name="stockQuote" maxOccurs="*" />
  </Element>

  <ElementType name="error">
    <element type="reason" />
  </ElementType>

  <ElementType dt:type="string" name="reason" />

</schema>

</contract>
Request for "getQuote"

M-POST /stockquotes HTTP/1.1
Host: amarg5:8586
Content-Type: text/xml
Man: "http://www.upnp.org/service-control/m-post"; ns=01
01-MethodName: getQuotes
01-MessageType: Call
Accept-Language: en-gb, en;q=0.8
Referer: http://amarg5/uPnPService/Services/Stock/Client/ticker.htm
Content-Length: 327
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Connection: Keep-Alive
...

```

FIG. 47

...  
<symbol>MSFT</symbol>  
Response for "getQuote"

HTTP/1.1 200 OK  
Connection: close  
Cache-Control: private  
Date: Mon Aug 16 15:37:35 PDT 1999  
Expires: Mon Aug 16 15:37:35 PDT 1999  
Content-Type: text/xml  
Content-Length: 7912  
Man: "http://www.upnp.org/service-control/m-post"; ns=01  
Ext:  
01-MessageType: CallResponse

<stockQuote>  
  <company>Microsoft%20Corporation</company>  
  <ticker>MSFT</ticker>  
  <previousClose>84%2011/16</previousClose>  
  <openingTrade>85%201/16</openingTrade>  
  <lastTrade>84%205/16</lastTrade>  
  <volume>28.66%20Mil</volume>  
</stockQuote>

001020"8TE95450

FIG. 48

<!-- XDR Schema for protocol section of contract -->

<schema name="contract"  
  xmlns="urn:schema-microsoft-com:xml-data"  
  xmlns:dt="urn:schema-microsoft-com:datatypes">

  <ElementType name="contract"  
    xmlns:protocolNS="contract-protocol"  
    xmlns:msgPatternNS="contract-msgPatterns"  
    xmlns:schemaNS="urn:schema-microsoft-com:xml-data">

    <element type="protocolNS:protocol" />

    <element type="msgPatternNS:RequestResponse" minOccurs="0"  
maxOccurs="\*" />

    <element type="msgPatternNS:SolicitResponse" minOccurs="0" maxOccurs="\*" />

    <element type="schemaNS:schema"           minOccurs="0" maxOccurs="\*" />

  </ElementType>

</schema>

...

FIG. 49

...

Protocol

<!-- XDR Schema for protocol section of contract -->

<schema name="contract-protocol"  
    xmlns="urn:schema-microsoft-com:xml-data"  
    xmlns:dt="urn:schema-microsoft-com:datatypes">

    <ElementType name="protocol">

        <!-- ID -->  
        <AttributeType name="id" dt:type="id" />  
        <Attribute type="id" />

        <group order="one">  
            <element xmlns:http="contract-protocol-HTTP" type="http:HTTP" />  
            <element xmlns:gena="contract-protocol-GENA" type="gena:GENA" />  
            // other protocol definitions go here  
        </group>

    </ElementType>

</schema>

...

1114660131696460



FIG.51

```
...

<ElementType name="HEADER">
  <attribute type="name" />
  <attribute type="value" required="yes" />
</ElementType>

<!-- Verb declarations -->
<ElementType name="GET"/>

<ElementType name="POST">
  <element type="PARAM" minOccurs="0" maxOccurs="*" />
</ElementType>

<ElementType name="PARAM">
  <element type="name" />
  <element type="default" />
  <element type="value" />
  <element type="required" />
</ElementType>

<AttributeType name="name" dt:type="string" required="yes" />
<AttributeType name="default" dt:type="string" />
<AttributeType name="value" dt:type="string" />
<AttributeType name="required" dt:type="boolean" default="no" />

<ElementType name="M-POST">
  <element type="MAN" />
</ElementType>

<ElementType name="MAN" dt:type="string" />

</schema>
```